Why can't AppLocale just be added to the Compatibility property sheet page?

devblogs.microsoft.com/oldnewthing/20100616-00

June 16, 2010

Raymond Chen

Commenter DoesNotMatter wants to know why AppLocale cannot just be added to the Compatibility property sheet as a dropdown option. One of the things about having a huge topic backlog is that if I just wait long enough, there's a good chance somebody else will answer it, and then I don't have to write anything. And more often than not, that somebody else is Michael Kaplan, who addressed this question in April 2010: Not only is AppLocale not installed by default, AppLocale does everything in its power to remind you that you shouldn't be using it! AppLocale is the emergency compact spare tire that you pull out of your trunk. Its job is to get you home, at which point you can fix the problem properly. You shouldn't be driving on your emergency compact spare as part of your normal daily routine. Why does changing the `CP_ACP` code page require a logoff/logon cycle? Because without it, you would have a Frankenstein configuration situation, where two programs think they're speaking the same language to each other, but aren't. This would happen if one program was launched before you changed the locale, and the other was launched after it. Sure, if the communication was done through the clipboard `CF_TEXT` data format, or via one of the system-defined window messages that contain strings, then the window manager can convert from one code page to the other (though it will have to round-trip through Unicode). But that's an awful lot of work for something that *isn't even a valid steady-state configuration*. And besides, it wouldn't even fix the other communication channels between processes, such as custom clipboard formats or private window messages. For example, suppose you have a copy of LitWare running, and then you change the locale, and then you run another copy of LitWare. You then drag an object from the first copy of LitWare to the second. If LitWare's custom data format uses ANSI strings, then the first copy will encode the object name using the old locale, and the second copy will decode it with the new locale. And since this is a custom data format, there's nowhere the window manager or OLE can step in and say, "Oh, wait, I know what you're doing. There's a string embedded in that structure. Let me convert it for you." I encounter this problem myself with a Chinese-language program I use. The program uses a Chinese code page rather than Unicode, and I have to use AppLocale to get it to display anything other than meaningless gibberish. (And even then, it still displays what appears to me to be mostly gibberish, but that part of the problem is my own fault for not knowing how to read very much Chinese.) I have to remember that I cannot copy/paste any Chinese characters into or out of the program because the result will be garbage due to the code page mismatch.

Mind you, you can just ignore the logoff reminder that appears when you change the default locale and continue running your Frankenstein configuration. Just understand that you’re now in a world where programs can no longer communicate with each other reliably.